International Journal of Hybrid Information Technology
Vol. 9, No.8 (2016), pp. 65-84
http://dx.doi.org/10.14257/ijhit.2016.9.8.07

Autonomous Migration of Virtual Machine Based on OpenStack

Li Zhou[1,2], Zhuoer Yu[1,2], Jilin Zhang[1,2,3,4,*], Jian Wan[1,2,4,5] and Yusen Wu[1,2]

[1]*School of Computer and Technology, Hangzhou Dianzi University, 310018, Hangzhou, China*
[2]*Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou, China*
[3]*College of Electrical Engineering, Zhejiang University, 310058, Hangzhou, China*
[4]*School of Information and Electronic engineering, Zhejiang University of Science & Technology, 310023, Hangzhou, China*
[5]*Zhejiang Provincial Engineering Center on Media Data Cloud Processing and Analysis, Hangzhou, Zhejiang, China*

Abstract

For the problem of compute nodes load unbalance leads to the overload node impact VM performance and the underload node waste electrical energy. Dynamic VM consolidation is an efficient approach for improving the utilization of servers and reducing energy costs. In this paper, we introduce dynamic consolidation into OpenStack and implement AMVM based on OpenStack. AMVM collect and analyze all performance data of VM and compute node in OpenStack. Then AMVM automatically schedules VM live migrate to the most appropriate node according to the real-time load of all compute nodes. Then we use hybrid-copy instead of pre-copy in AMVM to improve the efficiency of VM live migration. Finally, we analyze the performance of hybrid-copy, hybrid-copy can reduce page faults compared with post-copy and reduce total data transferred compared with pre-copy.

Keywords*: cloud computing; VM live migration; OpenStack; dynamic consolidation*

1. Introduction

With the further development of cloud computing, more and more enterprises provide public cloud by building a virtualized data center or build a business-oriented internal private cloud. The number of cloud computing data centers are increasing, the number of the servers also speed expansion of the scale. However, servers consume a lot of electrical energy. According to the United States department of energy's report, the power consumption of data center accounted for 1.5% of all US energy consumption and the demand for electrical energy is still at 12% per year rate of growth [1]. In addition, if the data centers of around the world as a "country", then the total energy consumption of this country rank No.15 in the world [2]. Furthermore, Global data center emissions of 11.6 million tons of carbon dioxide in 2007, and the carbon dioxide emissions of IT equipment accounted for 2% of global carbon dioxide emissions [3].

To realize energy efficiencies within cloud data centers, an efficient approach is dynamic consolidation. For underload node, VMs running on this node live migrate to another node, and shutdown the underload node to minimize the number of active nodes and economize the electric power resource. For overload node, VMs running on this node also live migrate to another node to avoid performance degradation experienced by the VMs.

ISSN: 1738-9968 IJHIT

VM live migration is one of the key technology to realize dynamic consolidation. VM live migration is to migrate a VM from a source node to another node as soon as possible within a short downtime. During the VM live migration, all transferred data include memory pages, VCPU state, device status, and network redirection. VM live migration is widely used in load balance of servers in data center [4, 5], online maintenance, fault-tolerant system and energy management [6]. Therefore, improving the performance of VM live migration is important to improve the resource utilization and reduce the energy consumption in cloud data center.

In this work, we introduce dynamic consolidation into OpenStack – a popular open-source cloud computing platform and realize autonomous migration of virtual machine (AMVM) in OpenStack. For the problem of VM uneven distribution of OpenStack, the AMVM will collect and analyze all performance data of VM and computing node in OpenStack. Then AMVM will automatically schedule VM live migrate to the most appropriate node according to the real-time load of all compute nodes. The AMVM is composed of three modules: performance monitoring module, VM migration scheduling module and disk array driver module. The performance monitoring module responsible for collect real-time performance data of all compute nodes and VMs in OpenStack. All performance data are saved in RRD database. The VM migration scheduling module analyze the gathered performance data to execute workload adjustment for computing node which triggered load threshold. The workload adjustment will migrate some VMs from underload or overload compute node to another appropriate compute node. Besides, the disk array driver module can help VM which mounts a volume live migrate to destination node successfully.

To further improve the efficiency of VM live migration, we use hybrid-copy in OpenStack instead of per-copy. Hybrid-copy is composed of three-phase. Firstly, all memory pages of VM will copy to the destination node in one round. Secondly, VM shutdown in a short time and transfers VCPU state and other necessary device state to the the destination node. Finally, VM restores running on the destination node. The remaining dirty memory will be synchronized through on-demand request and active push. Compared with the pre-copy, hybrid-copy can avoid iteratively copy dirty memory of VM. Compared with the post-copy, hybrid-copy effectively reduces the number of page fault of VM after VM restore running on destination node. Besides, we evaluate the performance of hybrid-copy when VM under a different workload. The result of experiment shows that hybrid-copy can reduce page faults compared with post-copy and reduce total data transferred compared with pre-copy.

The remainder of the paper is structured as follows, in the next section, we discuss the related work. Follow this, there is a section detailing the overall design and details of each module of AMVM. In section 4, we describe the hybrid-copy of VM live migration and experimentally evaluate the performance of hybrid-copy in different workload compared with pre-copy and post-copy. Finally, we conclude the paper in section 5.

2. Related Work

With the rapid development of cloud computing, the VM scheduling algorithm of cloud computing has also been a great progress. In order to reduce the energy consumption of data center, Beloglazov et al. [7, 8] proposed energy-aware allocation heuristics provision data center resources to client applications in a way that improves the energy efficiency of the data center, while delivering the negotiated Quality of Service (QoS). In addition, they put forward the VM optimal placement algorithm and VM selection algorithm that is to shutdown the needless servers to achieve energy saving.

In order to improve the system resources utilization, Fabien Hermenier et al. [9] proposed a resource scheduling algorithm named Entropy, this algorithm considers the

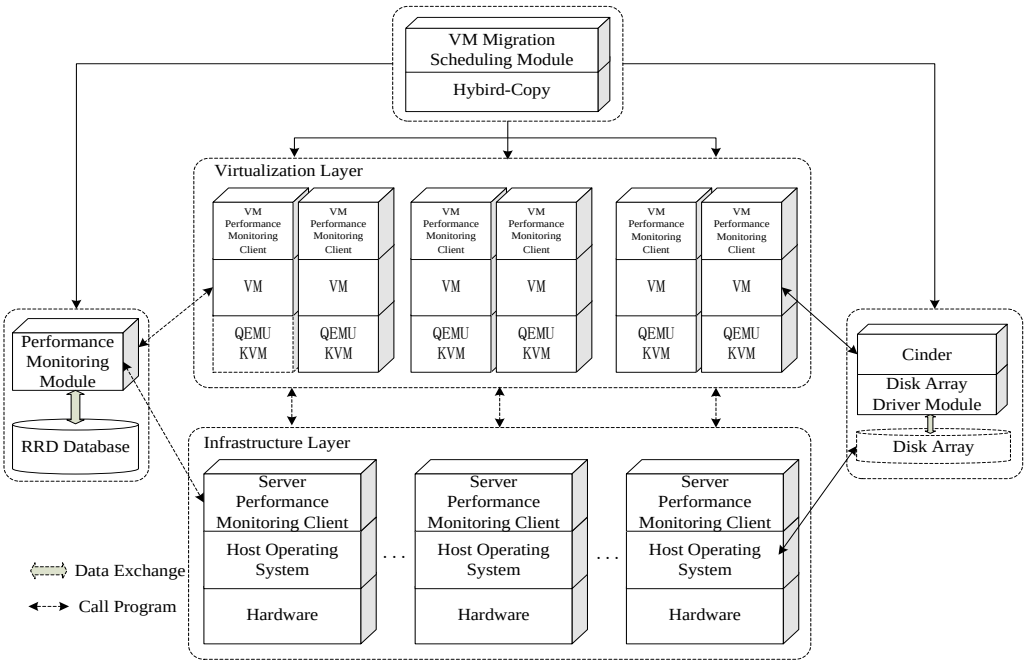

Figure 1. Architecture of AMVM

effects of two important factors on the reconfiguration time and the migration time of the VM. Hien Nguyen Van et al. [10, 11] proposed a dynamic scheduling algorithm to solve the problem of virtual resource scheduling in cloud computing.

Nathuji et al. [12, 13] proposed a set of management components of cluster layer and data center layer, and put forward the VM aware energy consumption budget to add multiple distributed manager into VirtualPower framework. The goal of VirtualPower is maximized the performance or utilization in a certain energy budget.

Jung et al. [14] proposed a framework Mistral that optimizes energy consumption and improves the performance gains, reduce overhead caused by caused by the various operations and the controller itself to maximizing the overall utilization.

Zhu et al. [15] presented three individual controllers that each operating at a different timescale, which places compatible workloads onto groups of servers, react to changing conditions by reallocating VMs.

VM live migration is a key feature of system virtualization technologies. Pre-copy [16] is a popular mechanism of live migration, which has been implemented in most hypervisors such as KVM [17], Xen [18] and VMware [19]. The main idea of pre-copy is that memory of VM is transferred to the destination node in a succession of iterations until the remaining dirty memory can be transferred in a short enough stop and copy phase which will not cause prolonged VM downtime.

Hines et al. [20, 21] propose using post-copy instead of pre-copy for live migration. In post-copy, the CPU and device state are transferred immediately to the destination node firstly, then the VM resume running on the destination node. The memory of VM will be synchronized to the destination node through on-demand request and active push. Compared with pre-copy, post-copy reduces the total migration time and total data transferred but lead to performance degradation of VM due to page faults which must get memory page from the source node.

Liu et al. [22] design and implement a CR/TR-Motion that adopts checkpointing/recovery and trace/replay technology to provide fast, transparent VM migration in LAN. Jin et al. [23] propose using adaptive compression of transferred data. The dirty page will be compressed before transferred to the destination node. Then compressed data will be decompressed and received. However, this approach increases the

system overload. Cerroni [24] proposes an analytical model that can evaluate the performance of an inter-datacenter network for federated clouds during multiple live migrations of VMs.

3. Autonomous Migration of Virtual Machine

The purpose of AMVM is to reduce energy costs and improve the server resource utilization. AMVM collect and analysis the performance data of all compute nodes and VM in OpenStack in real time. Then AMVM identifies the overload compute node and underload compute node to execute load adjustment. For overload nodes need to move part of the VM to reduce load, for underload nodes need to migrate all of the VM to other node and this node to be in the resting state automatically to reduce the electrical energy consumption.

Figure 1 show the architecture of AMVM. The infrastructure layer is composed of all servers and other hardware devices. The infrastructure layer provides the underlying computing, network and storage resources for the upper layer to achieve virtualization.

Besides, the infrastructure layer also is the key of AMVM, because the energy consumption of data center is mainly due to the hardware of the infrastructure layer. So the ultimate aim of AMVM is to improve the physical resource utilization by the VM scheduling. The virtualization layer is composed of KVM, QEMU, Libvirt and other basis software. Virtualization layer virtualizes CPU, memory and I/O devices of the infrastructure layer, then the virtualized resources are used by OpenSatck. Furthermore, AMVM is improved and optimized based on the OpenStack Kilo. In addition to OpenStack components, AMVM added the disk array driver module, performance monitoring module, and VM migration scheduling module.

3.1. Disk Array Driver Module

In OpenStack Cinder, all third party storage devices are required to implement their driver program. Then, this device can provide persistent storage volume services for VM as storage backend in the Cinder. So in the AMVM, we realize a driver of a disk array to achieve create, delete, mount the volume and other functions in the Cinder. Besides, to address the exception of VM live migration that mounts a volume, we realize a volume migration auxiliary program to help VM with volume live migrate to destination node normally.

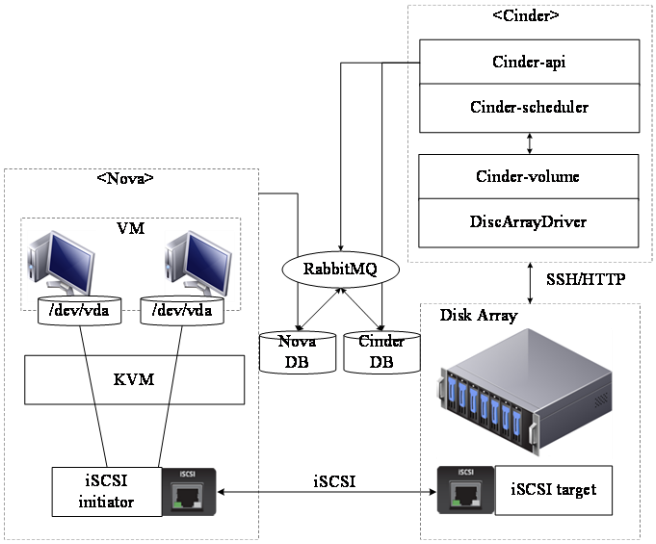

Figure 2. Internal Architecture of Cinder and Disk Array

3.1.1. Workflow of Disk Array Driver Module

Because the root partition of VM is not persistent, once this VM is deleted or failed, the data of VM cannot be restored. So VM need mount a block storage volume which provided by Cinder to realize persistence.

The internal architecture of Cinder and disk array as shown in Figure 2. The cinder-volume service running on the storage node to manage storage space of each storage node, so that several storage nodes may merge to form a storage resource pool to provide persistence storage service for OpenStack VM. We implement the DiscArrayDriver, so Cinder can use a disk array as storage backend. The volume of Cinder can be scheduling by cinder-scheduler service. All operations of volume by Cinder will eventually need to call DiskArrayDriver to achieve, and DiskArrayDriver sends commands to disk array through the SSH or HTTP to manage storage resources. The compute node in OpenStack and the disk array are interconnected by iSCSI, so that the block storage of disk array can be mapped to the compute node through iSCSI. The mapped block storage can be mounted to the specified VM as a volume by Libvirt to achieve persistent storage. DiskArrayDriver in the process of running need to read the configuration of the disk array and connection information in real time, so that the maintenance of the DiskArrayDriver does not need to restart the cinder-volume service and make the new configuration into effect.

Furthermore, the cinder-API service is responsible for receiving and processing all RESTful requests, and then send the request into RabbitMQ; cinder-scheduler service is responsible for handling tasks in the task queue, and according to the set of strategies to select an appropriate storage node to execute the task.

3.1.2. Volume Migration Auxiliary Program

When the VM with a volume to be migrated, the mapping relationship between VM and source node also have to turn into the mapping relationship between VM and destination node, then the VM can access storage volume on the destination node after live migration. However, the migration of volume state relies on the support of disk array driver. Therefore, AMVM realized the volume migration auxiliary program for VM and volume state live migration correctly. The workflow of volume migration auxiliary program as shown in Figure 3.

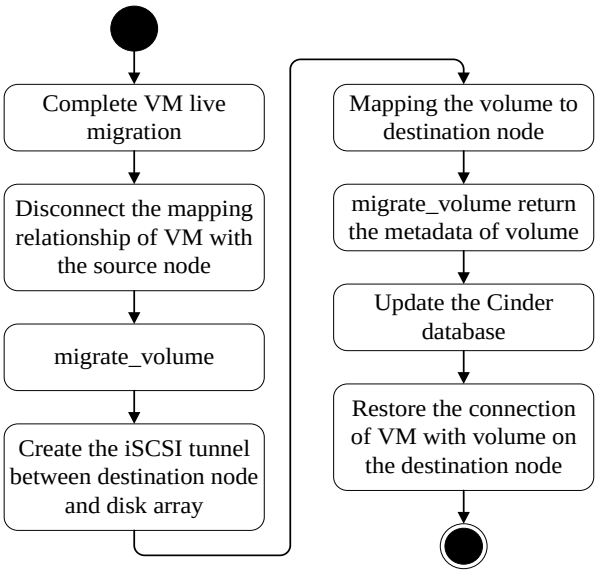

Figure 3. The Workflow of Volume Migration Auxiliary Program

At the beginning of the VM live migration, Nova will be calling Libvirt's API to execute VM migration. After VM have migrated to the destination node, the rest of VM state include volume state will be migrated to the destination node. For the migration of the volume state, the program disconnects the mapping relationship between VM and source node firstly and the volume is in the state of being unable to read and write. Secondly, the volume migration auxiliary program calls migrate_volume function, this function to determine whether the destination node for the first time to connect disk array. If so, the volume migration auxiliary program has to create the iSCSI tunnel between the destination node and disk array. Then, mapping the volume to destination node instead of the source node. Finally, the connection information of the volume with destination node will send to Cinder. Cinder need to update the new connection information to the database, and then Cinder restores the connection of VM and volume on the destination node to complete volume live migration.

3.2. Performance Monitoring Module

Performance monitoring module is responsible for collecting and storing the performance data of all servers and VM in OpenStack. This module includes server performance monitoring client, VM performance monitoring client, data collection server and RRD database. The server performance monitoring client and the virtual machine performance monitoring client are installed on each computing node, and data collection server and RRD database are installed on the control node. The architecture of performance monitoring module as shown in Figure 4.

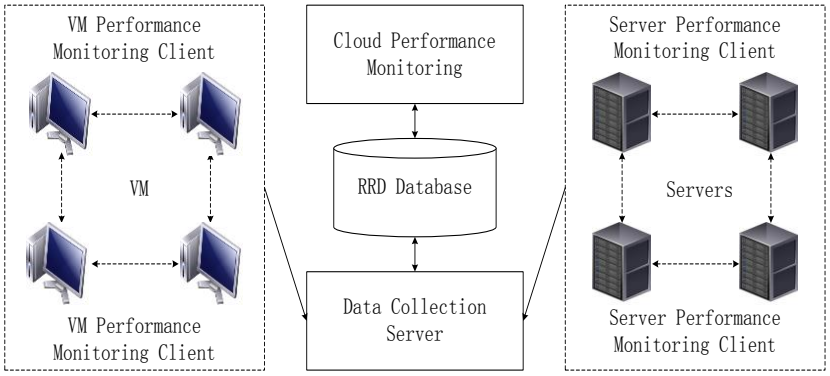

Figure 4. The Architecture of Performance Monitoring Module

Data collection server receives performance data from clients periodically, then save performance data to RRD database file respectively after data process. Data collection server can use unicast or multicast mode to collect performance data. In the unicast mode, the client needs to record the IP address and port of data collection server for send performance data to realize separate communication between each client and server across the network. In the multicast mode, client and server need in the same network and the client will send performance data to all nodes on the LAN. Therefore, the client will also receive performance data from other clients. The performance monitoring module uses unicast mode and the client send performance data to the server each 15 seconds.

Server performance monitoring client through Linux system interface to collect the performance data of compute nodes, but also use Nova's API to get the list of VM on the compute node. The VM performance monitoring client according to different client operating system use different API to collect the performance data of each VM. The specific performance indexes of server and VM are shown in Table I and II.

Table 1. Performance Index of Server

Performance Index	Description
os_name	Server operating system
uuid	UUID of server
heartbeat	Heartbeat from server
load_one/five/fifteen	Server load each 1/5/15 minutes
proc_total/run	Number of total/running processes
cpu_num	Number of CPU
cpu_usr/system/ide	User/system/idle CPU utilization
mem_total	Memory size
mem_free/shared/buffers/cached	Free/shared/buffers/cached memory
swap_total	Swap space size
swap_free	Free swap space
disk_total	Total disk space
disk_free	Free disk space
pkts_in/out	Server packets received/sent
bytes_in/out	Server bytes received/sent(B/s)

Table 2. Performance Index of Vm

Performance Index	Description
vcpu_num	Number of VCPU
vcpu_util	VCPU utilization ratio
vmem_total	Memory size
vmem_util	Memory utilization ratio
vdisk_total	Vdisk space
vdisk_free	Free vdisk space
vdisk_bytes_read/written	VM bytes read/writen (B/s)
vpkts_in/ out	VM packets received/sent
vbytes_in/out	VM bytes received/sent(B/s)

Performance data are saved in RRD (round-robin database). RRD is a fixed size recycled database, an RRD database file contains multiple RRA, each RRA includes a fixed number of data storage ring, if the number of data record exceeds the limit, the new data record will cover the eldest data record.

Compared to the traditional relational database for preserve all performance data, the relational database will occupy a large amount of disk space, RRD more frugal with disk space. In addition, because each RRA recording performance data is based on time frequency, RRD Tool can draw the performance data into curve graph conveniently, this graph can clearly show the real-time running state and specific performance data of OpenStack in the browser. It is to achieve a simple cloud monitoring easily.

Data collection server will according to the host-name of compute node to create the corresponding file directory. Each directory preserves the performance data of this compute node and VM located in this node. Using this file directory can avoid maintain the corresponding relationship between VM and located compute node in performance monitoring module. Therefore, the VM migration scheduling module get the performance data of compute node and corresponding VM.

3.3. VM Migration Scheduling Module

The VM migration scheduling module analyzes the performance data, calculate each computing node work load. For the compute node which loads above the upper threshold, part of VM located this node will be migrate to other destination nodes. In order to reduce the energy consumption of the data center, for the compute node which load is lower than the lower threshold, all VM will be migrated to other destination node. Then the underload node will be shutdown or transfer to sleep mode.

3.3.1. Load Threshold Setting

The upper load threshold is mainly based on the CPU, memory and network utilization of server, if any one of them exceeds the upper load threshold that the server belongs to overload node that need execute load adjustment. The load adjustment will migrate the VM from overload node to other nodes until a load of this node does not trigger the upper threshold. The workflow of load adjustment as shown in Figure 5.

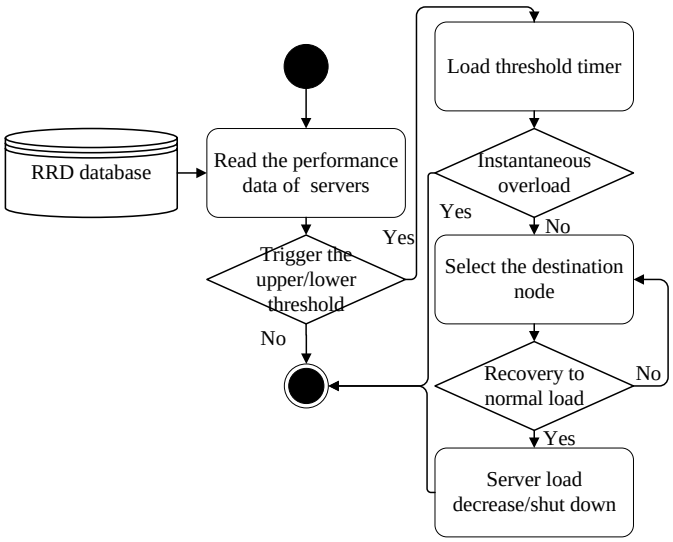

Figure 5. The Workflow of Load Adjustment

If the instantaneous load peak or valley of VM over the load threshold to adjust the load. It is easy to produce a lot of invalid migration operation, instead of decreasing resources utilization. Therefore, when a performance data of compute node exceeds the load threshold, VM migration scheduling module will continuously monitor this compute node. If this node keeps overload or underload within a period, then the load adjustment will be executed. Otherwise, if this node recovery to normal load in a short time that is not triggered load adjustment to avoid ineffective migration.

3.3.2. Selection Strategy of Destination Node

For the compute node, which need adjust load, we need to select the most appropriate node as the destination node of VM live migration in the remaining node. The inappropriate destination node will be likely to cause large amounts of load adjustment, so we need to select the best destination node through setting a series of strategies.

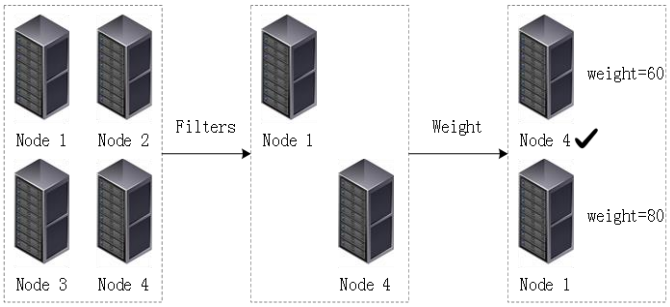

Figure 6. The Workflow of Select Destination Node

As shown in Figure 6, in OpenStack Nova Scheduler, the VM scheduling algorithm is composed of filter and weight. Using the filter to select a set of available compute nodes based on all kinds of indicators. The weight is to calculate a weight for each node in the set of available compute nodes and select the most appropriate node that is the lowest weight.

4. Hybrid Live Migration

The current existing VM live migration mechanisms mainly include pre-copy and post-copy. The pre-copy iterative copy dirty memory of the last round to the destination node, and then transfers the VCPU state to resume VM running on the destination node. The pre-copy will be caught in iterative copy for a long time and transfer many useless dirty pages when the dirty page generated faster than the speed of the network transmission. In contrast to the pre-copy, the post-copy copy the VCPU state before memory, and restore VM running on the destination node immediately. The VM memory will be synchronized through on-demand request and active push. Compared with the pre-copy, the downtime of post-copy is shorter than pre-copy and can guarantee that all of the memory pages will be transferred only once. However, the on-demand request gets the memory page from the source node through the network. Frequent on-demand request leads to the VM execution delay and the performance of VM decline.

In order to improve the efficiency of live migration of VM, we use the hybrid-copy replace pre-copy that is the default mechanism in OpenStck to implement VM live migration in AMVM.

4.1. Workflow

As shown in Figure 7, the workflow of hybrid-copy includes five phases.

1) Pre-Migration: Waiting for the VM migration command and select the appropriate destination node.

2) Reservation: It should ensure that there are enough physical resource on the destination node. The destination node will reserve those resource for migrated VM.

3) Full memory synchronization phase: All memory pages of VM sent to the destination node in one round, and create a dirty_bitmap for recording dirty page which is changed memory.

4) Stop copy phase: VM shutdown, the state of the VCPU, devices status and dirty_bitmap copy to the destination node.

5) Dirty page synchronization phase: VM restores running at the destination node, and synchronizes dirty page from the source node through two kinds of mechanism: on-demand request and active push.

Compared with the pre-copy, because hybrid-copy only transferred all memory of VM in memory synchronization phase, thus fundamentally avoid the situation of write memory frequently or dirty page produces greater than the speed of network leads to multiple iterative copy rounds. Compared with the post-copy, when VM resume running in the destination node, destination node has saved all memory of VM except for dirty page. Therefore most of the memory access of VM can obtain directly from the destination node's local memory rather than get from source node by network request. Thus, the performance cost of VM live migration will be drastically reduced the efficiency of application which running on VM is also improved.

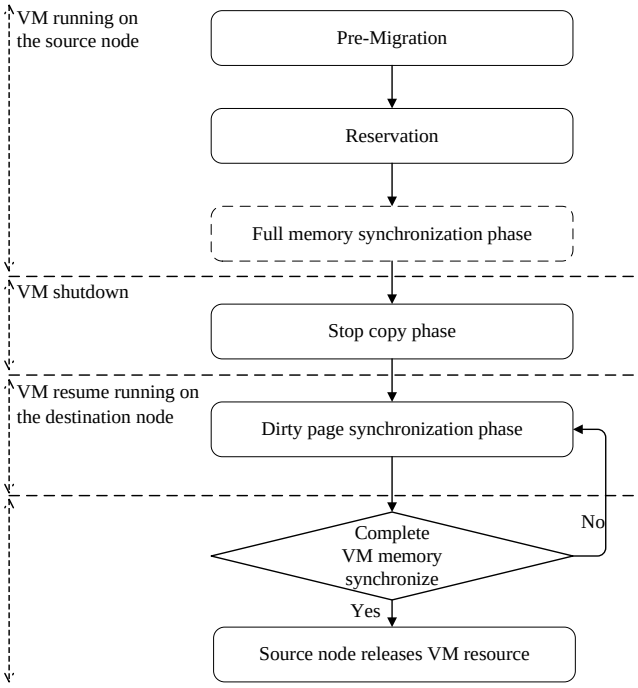

Figure 7. The Workflow of Hybrid-Copy

4.2. Dirty Page Synchronization

In the dirty page synchronization phase, VM uses on-demand request and active push to synchronize dirty page from the source node.

In the on-demand request, the destination node intercepts the page fault of VM and compared with the dirty_bitmap to confirm this page has not been synchronized, then sends this page fault to the source node. The source node receives this page fault and finds the corresponding memory page in local memory, then transfers this page to destination node after mark the corresponding bit in dirty_bitmap.

In order to further decrease page fault, the on-demand request adds prefetch page technique. Because of the order and the locality of the program, when VM generates a page fault that source node transfers corresponding memory page and neighbor memory page of this address to the destination node. When the VM access the next page will not trigger on-demand request again, VM can get this page from local memory, to enhance the efficiency of application running in VM.

In order to synchronize the remaining memory dirty page as soon as possible, hybrid-copy also use the active push mode. In active push mode, the source node sends dirty page to the destination node that has not been synchronized. If the corresponding bit of memory page in dirty_bitmap marked as 0, this page has already changed in the destination node, so discard this memory page, else receives it. In addition, if the source node receives an on-demand request in a process of active push, the source node interrupt current active push to fulfilling the on-demand request, as soon as possible to resume VM running from page fault exception.

4.3. Hybrid-Copy Model

The basic idea of hybrid-copy is, before by transfer all VM memory page and VCPU to the destination node in one round before VM resume running, and then synchronize the remaining dirty pages to reduce the number of the page fault. In this section, we describe

the hybrid-copy model and analyze the main parameters that affect the performance of live migration.

Table 3. Parameters of Hybrid-Copy Model

Parameters	Description
V_{mem}	VM memory size
V_{dirty}	VM dirty memory size
V_{mig}	Total data transferred
T_{down}	VM downtime
T_{mig}	Total migration time
T_{resume}	Duration of VM synchronize all memory page since resume running on the destination node
B	Network bandwidth between source node and destination node
D	Dirty rate of VM memory
F	Total page fault

We defined the key parameters and description of hybrid-copy in Table Ⅲ. In hybrid–copy, the dirty memory is produced in full memory synchronization phase. The time of this phase is $\frac{V_{mem}}{B}$, so the V_{dirty} as shown in equation (1) and the V_{mig} can be described as equation (2)

$$V_{dirty} = \frac{V_{mem} D}{B} \tag{1}$$

$$V_{mig} = V_{mem} + V_{dirty} = V_{mem} + \frac{V_{mem} D}{B} \tag{2}$$

The total migration time indicates that the VM start live migrate from the source node to resume running on destination node and all of VM memory has synchronized completely. So the total migration time includes the time of full memory synchronization phase, stop copy phase, dirty page synchronization phase. We sum the time of three phases as the following equation:

$$T_{mig} = \frac{V_{mem}}{B} + T_{down} + T_{resume} \tag{3}$$

In stop copy phase of hybrid-copy, source node needs to transfer the state of the VCPU, devices status and dirty_bitmap to the destination node. In addition, the destination node needs to load the VM device status to resume VM running. We use ε to describe the time overhead of VM resume running on the destination node. W_{VCPU} represent the size of VCPU. Therefore, the downtime can be calculated as the equation (4).

$$T_{down} = \frac{W_{VCPU}}{B} + \varepsilon \tag{4}$$

As for the T_{resume}, it represents the time required to synchronize all VM memory page completely since VM resume running on the destination node. Because there are two kinds of synchronization mechanism, the T_{resume} is the sum of time-consuming of the on-demand

request and active push mode. The parameter λ as the network delay between source node and destination node. So that the T_{resume} as describe as the following equation:

$$T_{resume} = \frac{V_{dirty}}{B} + F\lambda \tag{5}$$

In dirty page synchronization phase, the remaining dirty pages decrease with the active push mode, so the number of the page fault in unit time is also gradually reduced. We define the F as equation (6) where R is the access rate of VM memory and t is $\frac{V_{dirty}}{B}$.

$$F = \frac{\int_0^t \left(-\frac{V_{dirty}}{t} x + V_{dirty} \right) dx}{\int_0^t V_{mem} dx} tR = \frac{{V_{dirty}}^2 R}{2BV_{mem}} \tag{6}$$

So that we can combine equation (3) ~ (7) to define the total migration time as following:

$$T_{mig} = \frac{V_{mem} + W_{VCPU}}{B} + \frac{V_{mem} D}{B^2} + \frac{V_{mem} R D^2 \lambda}{2B^3} + \varepsilon \tag{7}$$

5. Performance Evaluation

In this section, we present a detailed evaluation of hybrid-copy compare with pre-copy and post-copy in different workload situation.

Our performance analysis the VM live migration in OpenStack. The OpenStack platform consists of a controller node and two compute nodes. The servers used for this experiment are Dawning A620r-G with Gigabit Ethernet interfaces and installed Ubuntu 14.04. The software includes QEMU 1.6.0, Libvirt 1.2.18. The operating system of VM is CentOS 6.5 and each VM with two vCPUs, memory size range from 256 MB to 4 GB.

In order to analyze the performance of all kinds of VM live migration mechanisms, we use the following four performance metrics as the evaluation standard:

1) Total migration time: The source node receives the VM live migration command until VM resumes running on the destination node. In addition, all relevant data of the VM has been completely synchronized to the destination node. This performance metric determines the occupation time of all kinds of physical resources during VM migration.

2) Downtime: The downtime is caused by synchronous VCPU state and other necessary device information. The VM can not provide service during the downtime.

3) Total data transferred: In the process of VM love migration, all data of the source node transferred to the destination node, which mainly includes memory pages, VCPU state and other devices status such as the network. This performance metric determines the utilization of network resources during the VM live migration.

4) Page faults: In the dirty page synchronization phase of post-copy and hybrid-copy, VM in the destination node to resume running. Due to the memory pages of VM are not completely synchronized in the destination node, the VM have to get memory page from the source node through the network. This performance metric is the total number of on-demand request. It reflects the performance loss caused by the VM live migration.

5.1. Empty Workload

In the empty load experiment, VM with different memory size immediately lives migrate to another node after power on. The experiment results as shown in Figure 8.

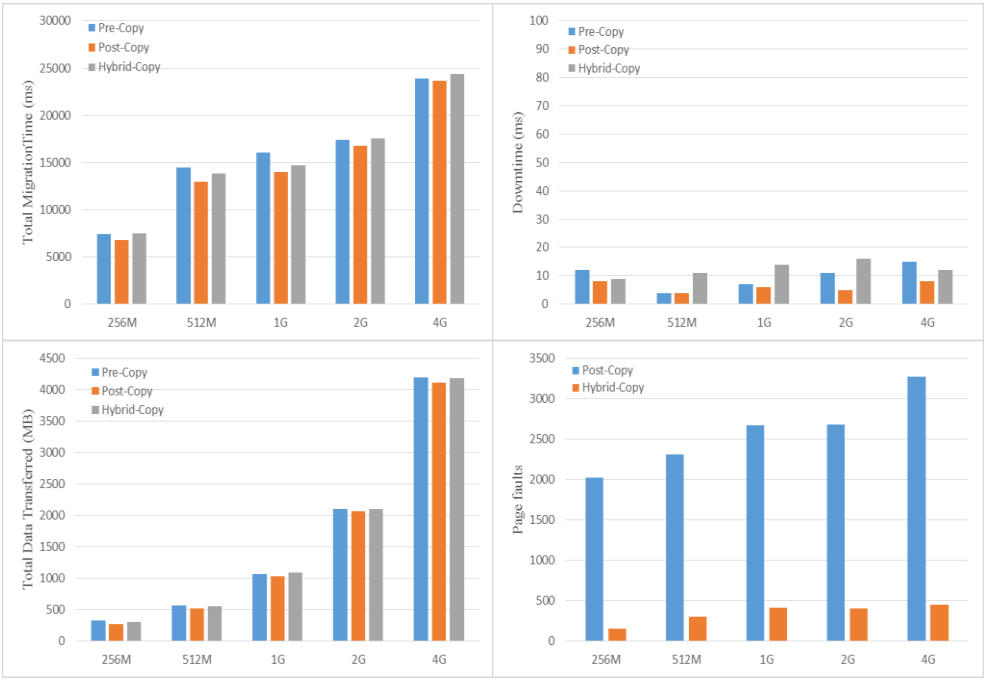

Figure 8. Overall Performance of Empty Workload

Because read/write memory operation of VM is very low under empty load, so the amount of dirty pages generated is also very small. In the empty load experiment, pre-copy is able to quickly end the memory iterative copy phase and enter the stop copy phase. The post-copy mainly relies on active push in dirty page synchronization phase. Hybrid-copy can synchronize the vast majority of the memory page in full memory synchronization phase. Therefore, the total data transferred of three kinds of migration mechanism is broadly equivalent to the memory size of VM. Besides, the total migration time is no big performance difference between all kinds of migration mechanism.

In pre-copy, because all memory pages have already saved at the destination node when VM resume running. So that there is no page fault in pre-copy. As the operation of access memory is less when VM under empty workload, the number of page faults in post-copy and hybrid-copy are very low. But compared to the post-copy, the hybrid-copy can get the majority of the memory access from the local memory of destination node, so the number of pages faults are less than 500 times.

5.2. Memory-Bound Workload

We use V8-benchmark programs on the VM to evaluate the memory profiling. V8-benchmark is a suite of JavaScript-based benchmarks developed by Google, and it includes DeltaBlue, Crypto, and Splay *etc*. Each benchmark is about two seconds, so we repeat to execute the program until VM live migration complete.

As shown in Figure 9, V8-benchmark as a memory intensive application will continue to execute write operations to generate memory dirty page. Therefore, pre-copy executes multiple rounds of iterative copy memory before stop copy phase, caused by the total migration time and total data transferred dramatically higher than other two kinds of VM live migration mechanisms. When the VM with 4GB memory, the total data transferred of pre-copy as high as 8869MB. The multiple rounds of memory copy in pre-copy leads to

the total migration time much higher than the other two mechanisms. The total migration time of pre-copy averagely is 208% of the hybrid-copy and 277% of the post-copy.

For the hybrid-copy, due to the dirty memory page also increased under the memory-bound workload. When the VM with 4GB memory, the total data transferred of hybrid-copy is 4716MB. Compared to the 4104MB physical memory of VM, the dirty page is 612MB. This also makes the total migration time of hybrid-copy is higher than post-copy in memory-bound workload.

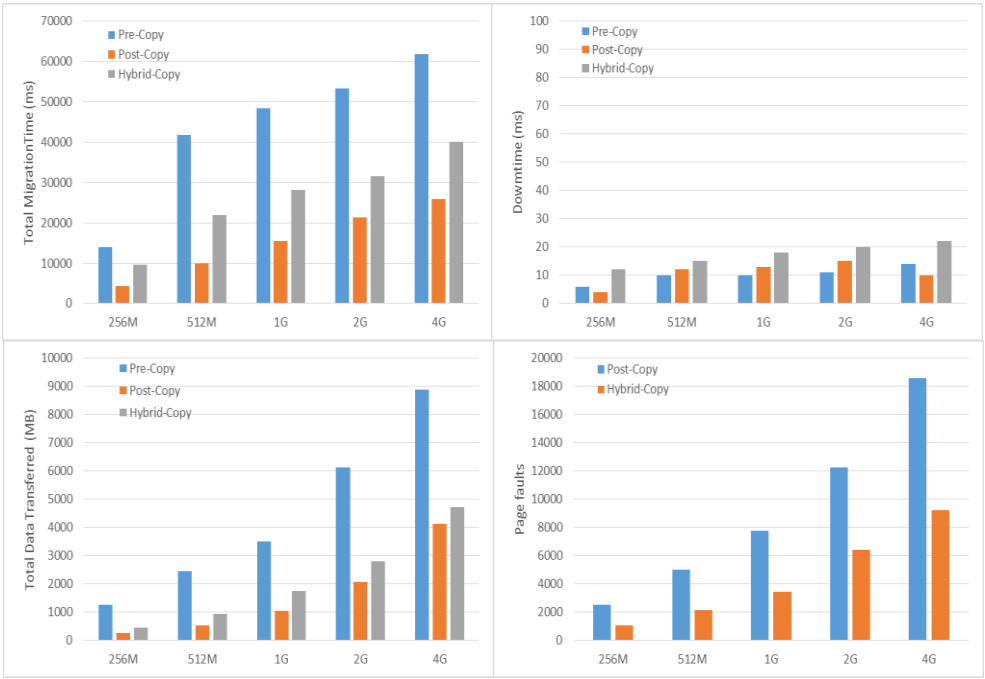

Figure 9. The Overall Performance of Memory-Bound Workload

As shown in the experimental results, even in the case of high memory load, the downtime of three kinds of VM lives migration mechanisms is basically in 20ms. Therefore, regardless of each VM live migration mechanism, the downtime of VM is very short and it can meet the requirement of the performance of the most applications.

However, in the memory-bound workload environment, the rate of the dirty page are increased, the page fault of hybrid-copy and post-copy are also increased. Compared to the post-copy, hybrid-copy significantly reduced the number of page faults. In the four group experiments, the total number of page faults is 5562 times in hybrid-copy, and is 11528 times in post-copy. This means hybrid-copy can reduce 51.8% page faults than post-copy. Reducing the number of page faults means that the more memory requests of a application running in the VM can be executed in the local memory of the destination node. The performance loss of VM caused by post-copy also can be reduced.

5.3. CPU-bound Workload

This experiment is to test the performance of the three kinds of VM live migration mechanisms under the CPU-bound workload. To simulate the high CPU load, we use VM to play video in this experiment.

As shown in Figure 10, the average total migration time of hybrid-copy is 66.5% of pre-copy and is 162.3% of post-copy. Due to the dirty page rate in CPU-bound workload experiment is relatively lower than memory-bound workload experiment. In the experiment of VM with 4GB memory, the total data transferred of pre-copy is 6769MB. Compared

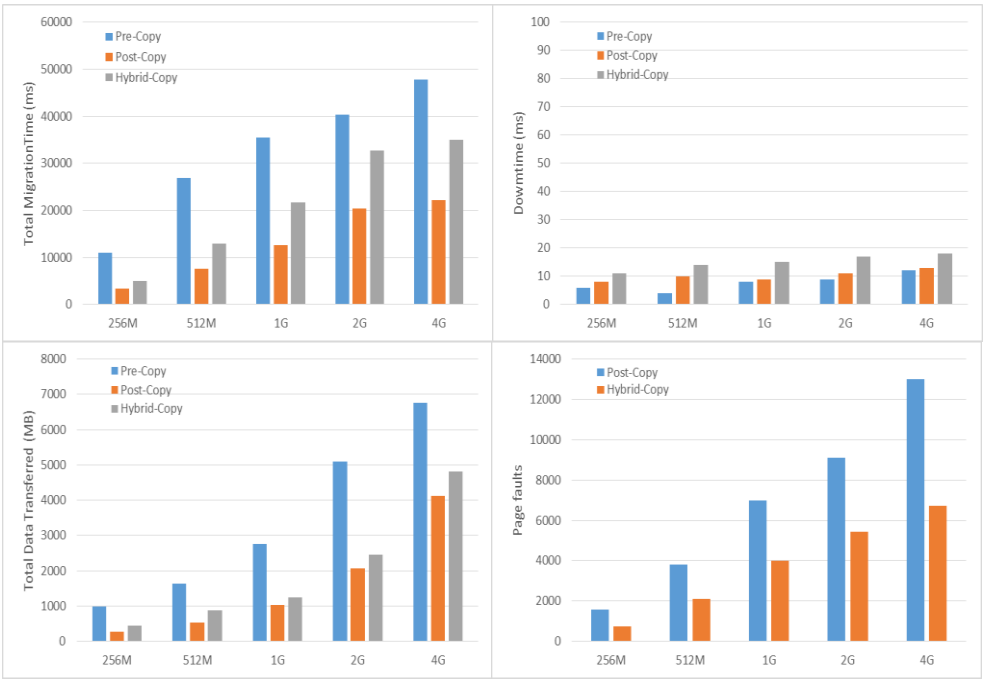

Figure 10. Overall Performance of CPU-Bound Workload

to the memory-bound workload, it is reduced 2100MB. However, it still higher than hybrid-copy and post-copy. The total data transferred of pre-copy is 140% of hybrid-copy and 164% of post-copy.

For hybrid-copy, the downtime of hybrid-copy is slightly higher than other two kinds of VM live migration mechanisms. The total data transferred of hybrid-copy is 123.1% of post-copy, but hybrid-copy reduced 45% page fault than post-copy averagely. Therefore, it is obvious that hybrid-copy has advantage in reducing performance loss caused by VM live migration under high CPU workload.

5.4. I/O-bound Workload

High I/O workload is a common scene in cloud computing environment. In the experiment, we use Bonnie++ to simulate the high I/O workload of VM. Bonnie++ is a set of well-known free benchmark tool set that can be used to test the performance of hard disk and file system. In this experiment, the test command is bonnie++ -D -x 10 -u root. The parameter D indicates that the way of read and write disk is direct I/O pattern. The parameter of -x 10 represents we repeat 10 times of test to ensure that high I/O workload during the VM live migration process.

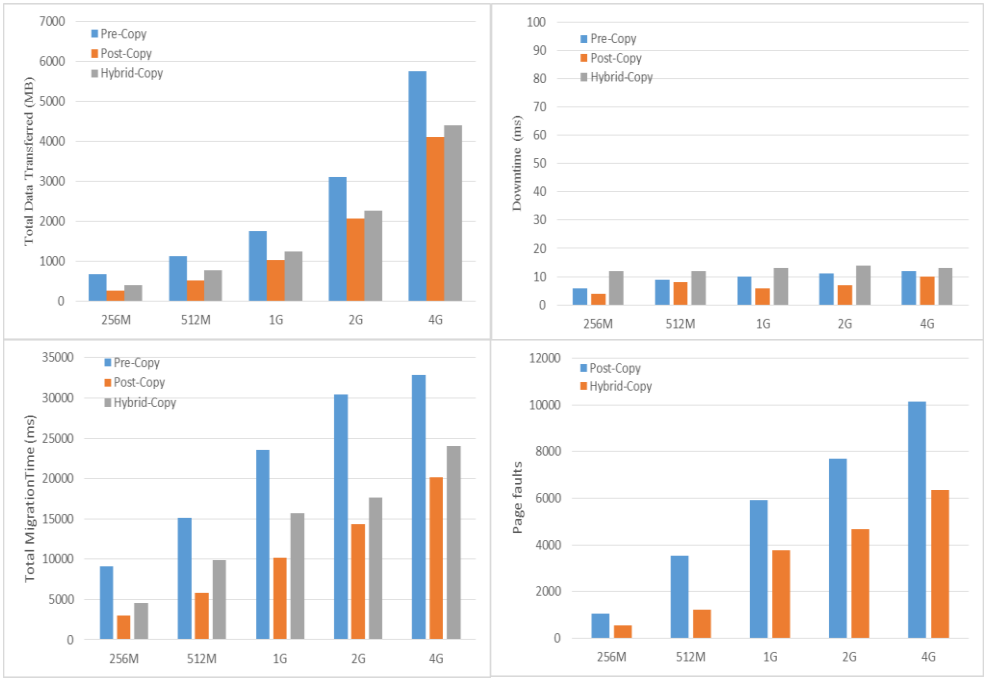

Figure 11. Overall Performance of I/O-Bound Workload

The experiment results as shown in Figure 11. As we use Network File System (NFS) between the source node and the destination node. The I/O operation of VM will not be interrupted when VM live migration except for the stop copy phase. Because the memory read/write rate of VM is low in high I/O workload experiment, the average dirty memory of the three kinds of VM live migration mechanisms respectively is 897MB, 11.5MB and 230.6MB. That is, the hybrid-copy requires to transfer 219.1MB memory data more than post-copy. But the hybrid-copy can reduce 41.7% page faults than post-copy in high I/O workload experiment. Therefore, the hybrid-copy can effectively reduce the number of page faults under the condition of increasing some transferred data.

6. Conclusions

In this work, we implement AMVM which automatically scheduling VM live migrate to the most appropriate node according to the real-time load data of all compute nodes to realize autonomous migration of VM in OpenStack. The AMVM includes the disk array driver module, performance monitoring module, and VM migration scheduling module.

To improve the efficiency of VM live migration in AMVM, we use hybrid-copy instead of pre-copy. The hybrid-copy transfers all memory page at first to ensure the destination node saves memory page of VM when VM resume running. Then the hybrid-copy synchronizes dirty page from a source node through on-demand request and active push. At last, we evaluate the performance of hybrid-copy when VM under the different workload. The result of experiments shows that hybrid-copy can reduce page faults compared with post-copy and reduce total data transferred compared with pre-copy.

Acknowledgments

This work is partly supported by the Ministry of Science and Technology-tube detection technology under Grants No.2014BAK14B04, the National High Technology Research and Development Program of China under Grant No.2015AA01A303, the Zhejiang Natural Science Funds under Grants NO.LY16F020018 and NO.LY13F020047, the National Natural Science Foundation of China under Grants No. 61572163, No.

61472112, NO.61572163, NO. 61202094 and NO.61472109, the National High Technology Research and Development Program of China under Grant No.2015AA01A303, Key Laboratory of Complex Systems Modeling and Simulation program, Ministry of Education and Chinese Postdoctoral Science Foundation No.2013M541780 and No.2013M540492.

References

[1] A. Hooper, "Green computing", Communication of the ACM, vol. 51, no. 10, **(2008)**, pp. 11-13.
[2] G, Cook and J. Van Horn, "How dirty is your data? A look at the energy choices that power cloud computing", Greenpeace, **(2011)**.
[3] G. Cook, "How clean is your cloud? Report, Greenpeace International", **(2012)**.
[4] L. Chen and H. Choi, "Approximation algorithms for data distribution with load balancing of web servers", In cluster: IEEE, **(2001)**, pp. 274.
[5] H. Kameda, J. Li and C. Kim, "Optimal load balancing in distributed computer systems", Springer Science & Business Media, **(2012)**.
[6] B. Nicolae and F. Cappello, "A hybrid local storage transfer scheme for live migration of i/o intensive workloads", Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing: ACM, **(2012)**, pp. 85-96.
[7] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", Future generation computer systems, vol. 28, no. 5, **(2012)**, pp. 755-768.
[8] R. Buyya, A. Beloglazov and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges", arXiv preprint arXiv:1006.0308, **(2010)**.
[9] F. Hermenier, X. Lorca and J. M. Menaud, "Entropy: a consolidation manager for clusters", Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. ACM, **(2009)**, pp. 41-50.
[10] H. N. Van, F. D. Tran and J. M. Menaud, "SLA-aware virtual resource management for cloud infrastructures", Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on. IEEE, vol. 1, **(2009)**, pp. 357-362.
[11] H. Nguyen Van, F. Dang Tran and J. M. Menaud, "Autonomic virtual resource management for service hosting platforms", Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. IEEE Computer Society, **(2009)**, pp. 1-8.
[12] R. Nathuji and K. Schwan, "Virtual Power: coordinated power management in virtualized enterprise systems", ACM SIGOPS Operating Systems Review. ACM, vol. 41, no. 6, **(2007)**, pp. 265-278.
[13] R. Nathuji, K. Schwan and A. Somani, "VPM tokens: virtual machine-aware power budgeting in datacenters", Cluster computing, vol. 12, no. 2, **(2009)**, pp. 189-203.
[14] G. Jung, M. Hiltunen and K. R. Joshi, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures", Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on. IEEE, **(2010)**, pp. 62-73.
[15] X. Zhu, D. Young and B. J. Watson, "1000 islands: Integrated capacity and workload management for the next generation data center", Autonomic Computing, 2008. ICAC'08. International Conference on. IEEE, **(2008)**, pp. 172-181.
[16] C. Clark, K. Fraser and S. Hand, "Live migration of virtual machines", Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. USENIX Association, **(2005)**, pp. 273-286.
[17] A. Kivity, Y. Kamay and D. Laor, "kvm: the Linux virtual machine monitor", Proceedings of the Linux Symposium, vol. 1, **(2007)**, pp. 225-230.
[18] P. Barham, B. Dragovic and K. Fraser, "Xen and the art of virtualization", ACM SIGOPS Operating Systems Review, vol. 37, no. 5, **(2003)**, 164-177.
[19] M. Nelson, B. H. Lim and G. Hutchins, "Fast Transparent Migration for Virtual Machines", USENIX Annual Technical Conference, General Track, **(2005)**, pp. 391-394.
[20] M. R. Hines, U. Deshpande and K.Gopalan, "Post-copy live migration of virtual machines", ACM SIGOPS operating systems review, **(2009)**, vol. 43, no. 3, pp. 14-26.
[21] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning", Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. ACM, **(2009)**, pp. 51-60.
[22] H, Liu, H. Jin and X. Liao, "Live migration of virtual machine based on full system trace and replay", Proceedings of the 18th ACM international symposium on High performance distributed computing. ACM, **(2009)**, pp. 101-110.

[23] H. Jin, L. Deng and S. Wu, "Live virtual machine migration with adaptive, memory compression", Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on. IEEE, **(2009)**, pp. 1-10.
[24] W. Cerroni, "Multiple virtual machine live migration in federated cloud systems", Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on. IEEE, **(2014)**, pp. 25-30.

Authors

Li Zhou, she received her Master Degree from Hangzhou Dianzi University, Hangzhou, China, in 2003. She is currently an associate professor in School of Computer Science and Technology, Hangzhou Dianzi University. Her current research interests include virtual storage system, cloud storage, cloud computing and high performance computing.

Zhuoer Yu, he is a software developement engineer at NetEase, Inc. He received his Ms from Hangzhou Dianzi University in China, 2016. His research interests include cloud computing, distributed monitoring systems and performance of cloud systems.

Jilin Zhang, he received the PhD degree in Computer Application Technology from University of Science Technology Beijing, Beijing, China, in 2009. He serves as an assistant professor of software engineering in Hangzhou Dianzi University, China. His research interests include High Performance Computing and Cloud Computing.

Jian Wan, he received the PhD degree in Computer Application Technology from Zhejiang University, Zhejiang,China, in 1989. He is currently a professor in software engineering in Hangzhou Dianzi University,China. His research interests include Grid Computing, Service Computing and Cloud Computing.

Yusen Wu, he is now M.S. in School of Computer Science and Technology in Hangzhou Dianzi University, China. His research interests include Parallel Computing, High Performance Computing and Cloud Computing.